Timers and Interrupts

Anurag Dwivedi

ELECTRONICS
IITKANPUR CLUB

LET US REVISE

MCU

A small computer integrated in a single IC

MCU

A small computer integrated in a single IC

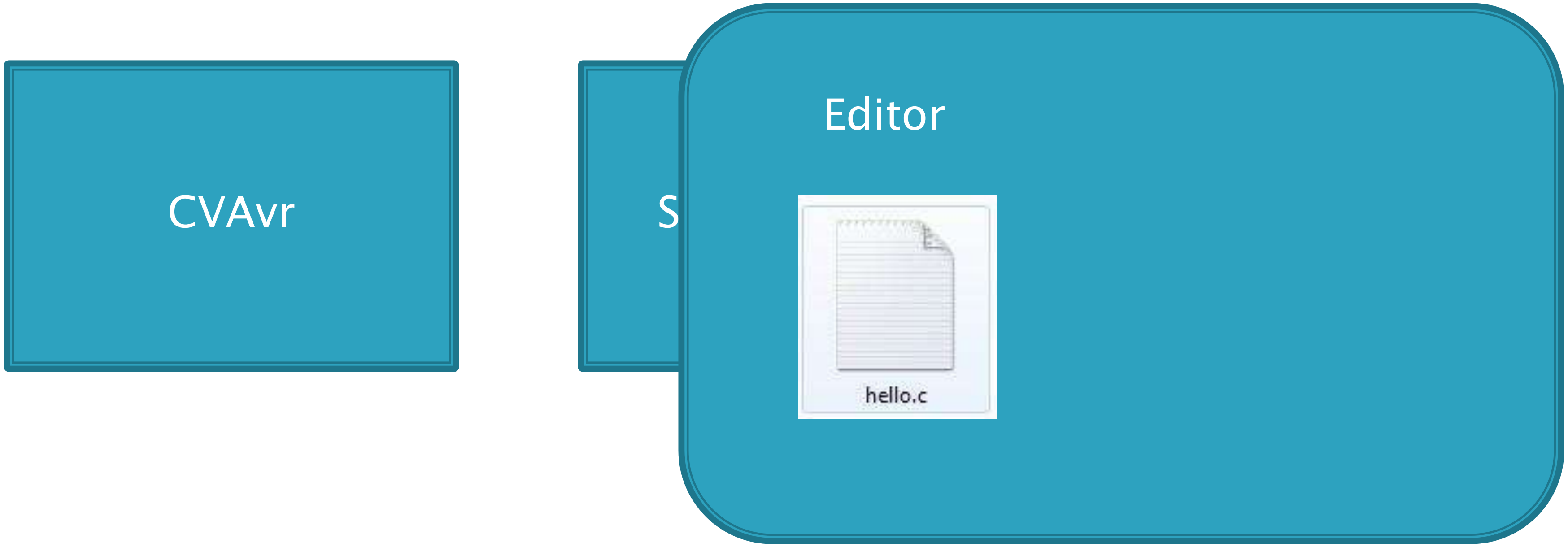
MCU

Has I/O pins, RAM and Memory

Software Used

CVAvr

Software Used



CVAvr

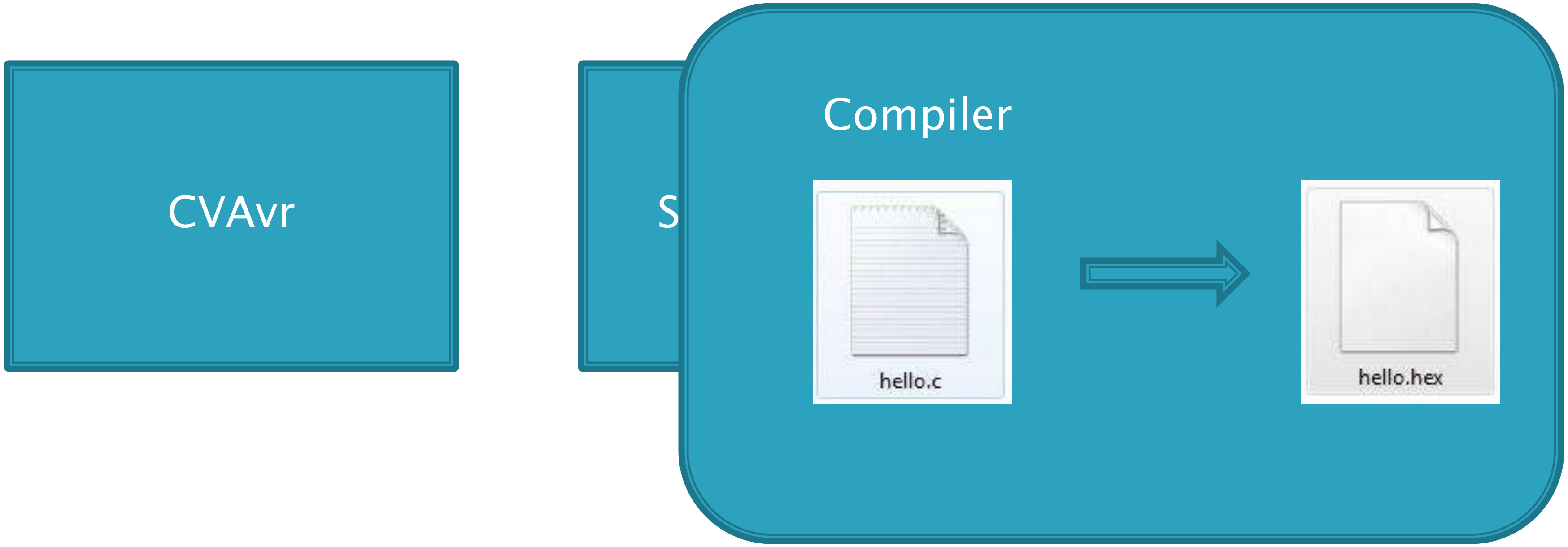
S

Editor

hello.c

CVAvr

Software Used



CVAvr

S

Compiler

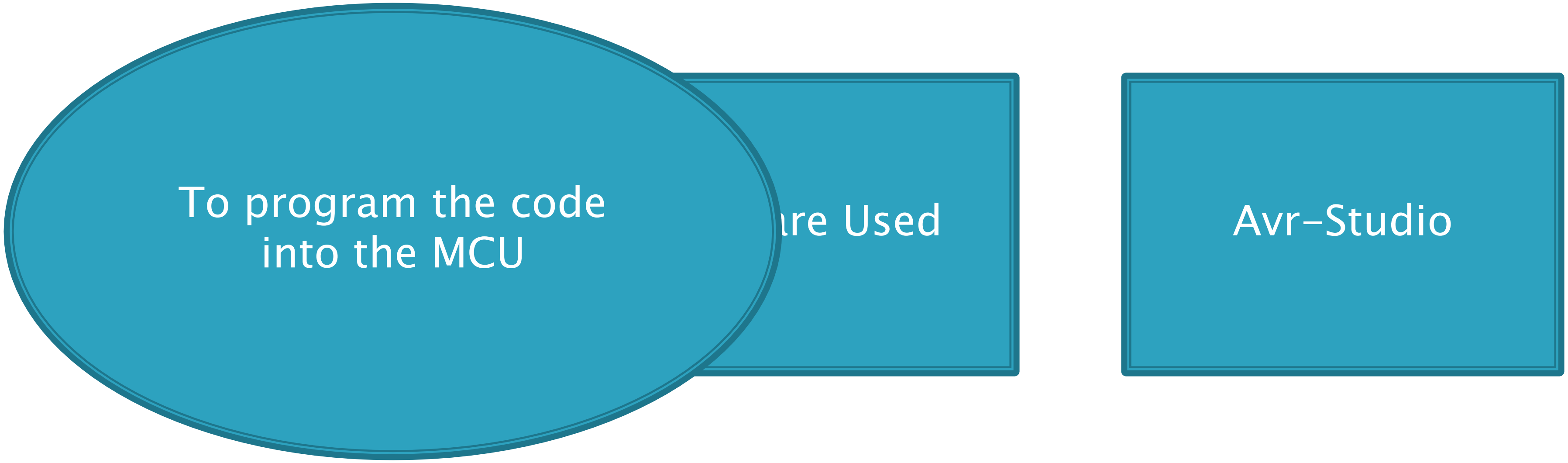
hello.c

hello.hex

Software Used

Software Used

Avr-Studio



To program the code into the MCU

are Used

Avr-Studio

MCU Coding

The data direction is set through DDR Register

MCU Coding

The data direction is set through DDR Register

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0

Function	Output	Output	Input	Output	Input	Input	Input	Output
DDRB	1	1	0	1	0	0	0	1

Value	High(+5V)	High(+5V)	Low(0V)	Low(0V)	Low(0V)	High(+5V)	High(+5V)	Low(0V)
PORTA	1	1	0	0	0	1	1	0

MCU Coding

MCU Coding

I/O ports are accessed by PORT and PIN Registers

```
.
.
.
While(1){
            PORTA.1 = 1; //sets the pin to 5V
            PORTA.1 = 0; // sets the pin to 0V

            X = PINA.0;  //reads the value of pin
                         // and copies it to X
}
.
.
.
```

I/O ports are accessed by PORT and PIN Registers

REGISTERS

- Registers are actual hardware memory locations inside the μC.
- What do we mean by this??
- Consider a 8-bit long register. Each bit of the register can be realized as a flip-flop.
- Ex. PORTX is a register.
- When you set the value of PORTA = 0X01, you physically set the corresponding flip-flop a value of +5 Volts.

ELECTRONICS CLUB IITKANPUR

TIMERS

- A Timer is usually a 8–bit register.
- It starts with

0	0	0	0	0	0	0	0	0

.

.

.

.

1	1	1	1	1	1	1	1	255

TIMERS

- 8-bit register.
- Values starts from 0 and goes up to 255.

TIMERS

- 8-bit register.
- Values starts from 0 and goes up to 255.
- Timer value increases by 1,after each period.

t = 0 T	0	0	0	0	0	0	0	0

TIMERS

- 8-bit register.
- Values starts from 0 and goes up to 255.
- Timer value increases by 1,after each period.

t = 1 T	0	0	0	0	0	0	0	1

TIMERS

- 8-bit register.
- Values starts from 0 and goes up to 255.
- Timer value increases by 1,after each period.

t = 2 T	0	0	0	0	0	0	1	0

TIMERS

- 8-bit register.
- Values starts from 0 and goes up to 255.
- Timer value increases by 1,after each period.

t = 255 T	1	1	1	1	1	1	1	1

TIMERS

- 8-bit register.
- Values starts from 0 and goes up to 255.
- Timer value increases by 1,after each period.

t = 255 T	1	1	1	1	1	1	1	1

- When the timer reaches its maximum value, in the next cycle, its value becomes 0 again and the process repeats itself.

TIMERS

- 8-bit register.
- Values starts from 0 and goes up to 255.
- Timer value increases by 1,after each period.

t = 256 T	0	0	0	0	0	0	0	0

- When the timer reaches its maximum value, in the next cycle, its value becomes 0 again and the process repeats itself.

TIMERS

- 8-bit register.
- Values starts from 0 and goes up to 255.
- Timer value increases by 1,after each period.

t = 256 T	0	0	0	0	0	0	0	0

- When the timer reaches its maximum value, in the next cycle, its value becomes 0 again and the process repeats itself.
- The timer frequency can be factors of the base frequency of the MCU.

TIMERS

- 8-bit register.
- Values starts from 0 and goes up to 255.
- Timer value increases by 1,after each period.

t = 256 T	0	0	0	0	0	0	0	0

- When the timer reaches its maximum value, in the next cycle, its value becomes 0 again and the process repeats itself.
- The timer frequency can be factors of the base frequency of the MCU.
- This process is **independent** of the CPU.

Simple Statistics

- Maximum value of timer is ***n*** and clock period is ***t***, then:

1. Timer period = t
2. Timer cycle period $= (n+1)\times t$
3. Frequency of timer (f) $= 1/t$
4. Frequency of timer cycle $= 1/(n+1)\times t$

Topics Covered so far…

- ✓ Registers
- ✓ Timers

Interrupts

- Interrupts means causing a break in a continuing process.

Why interrupts?

- Suppose you need to check for a condition A while running another condition B

- Simple Solution..

- Simple Solution..

```
while(1){
---- -> if (Event A == true)
---- -> // print event A has occurred
----
----
---- -> Event B
----


----
}
```

- Simple Solution..

```
while(1){
---- -> if (Event A == true)
---- -> // print event A has occurred
----
----
---- -> Event B
----


----
}
```

Do you see the problem in this approach??

- Simple Solution..

```
while(1){
---- -> if (Event A == true)
---- -> // print event A has occurred
----
----
---- -> Event B
----
---- -> Suppose Event A happens here
----
}
```

A Better Solution

We execute the event B in the normal way, in the while(1) loop.

We execute the event B in the normal way, in the while(1) loop.

```
.
.
while(1){
---
---
EVENT B
---
---
}
.
```

We consider the occurrence of event A as a interrupt

```
.
.
while(1){
---
---
EVENT B
---
---
}
.
```

We consider the occurrence of event A as a interrupt

```
.
.
while(1){
---
---
EVENT B
---
---
}
.
```

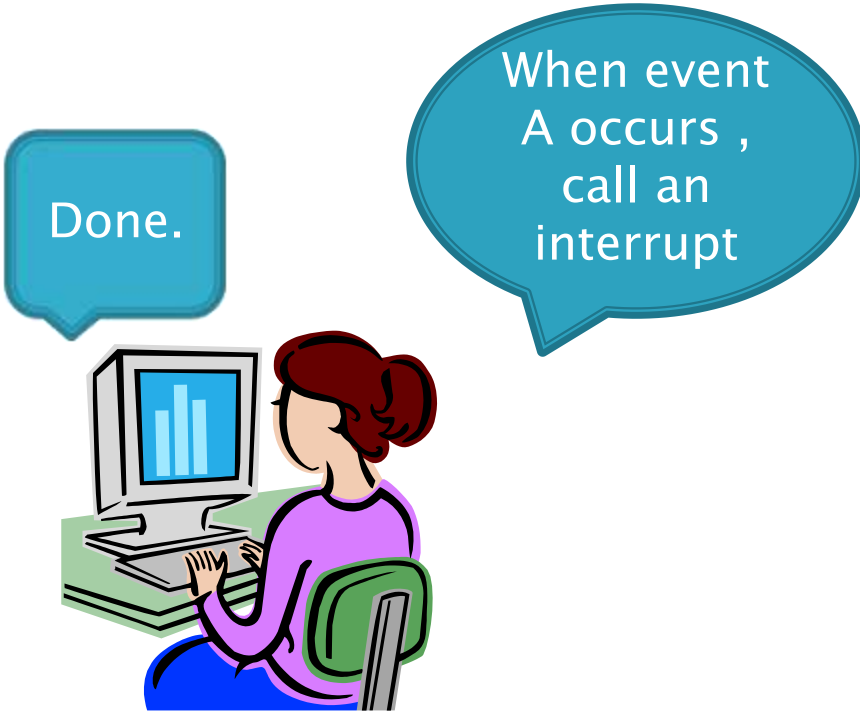
When event A occurs , call an interrupt

We consider the occurrence of event A as a interrupt

```
.
.
while(1){
---
---
EVENT B
---
---
}
.

handleA(){
.

}
```



We execute the required code in the handler of event A.

```
.
.
while(1){
---
---
EVENT B
---
---
}
.

handleA(){
.

}
```

We execute the required code in the handler of event A.

```
.
.
while(1){
---
---
EVENT B
---
---
}
.

handleA(){
.
// print event A has occurred
}
```

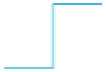
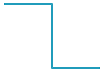
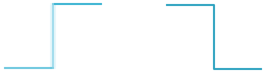
The BIG Question..

More on Interrupts

- Interrupts are special events that can “interrupt” the normal flow of a program.
- Whenever an Interrupt is called, the processor stops the normal program, handles the interrupt, and then resumes its normal work.
- There are two types of interrupts:
- External and Internal

External Interrupts

- The controller monitors the input at the special pins INT0 and INT1, whenever external interrupt is set on.
- We can configure the program to call an external interrupt whenever any of the following conditions are met.

- Rising Edge 
- Falling Edge 
- Any change 
- Low level

Topics Covered so far…

- ✓ Registers
- ✓ Timers
- ✓ Interrupts
- ✓ External Interrupts

Internal Interrupts

- The internal interrupts are called when different specific conditions are met by the timer value.
- This brings us to the next topic..

Timers and Interrupts

- Timers can generate certain interrupts: two, to be precise.
- These are called OVERFLOW interrupt and COMPARE MATCH interrupt.

OVERFLOW INTERRUPT

- An overflow interrupt is generated when the timer exceeds its maximum value and resets to 0

- The interrupt may or may not have a handler. In either case, the timer continues to run; remember: timers are **independent** of the CPU.

OVERFLOW STATISTICS

- Suppose a timer of maximum value ***n*** has a time period ***t*** (also called as clock period).

- Then :

  1. Timer cycle frequency $= 1/(n+1)\times t$
  2. OVERFLOW interrupt frequency $= 1/(n+1)\times t$

- If OVERFLOW interrupt is enabled, then an interrupt is generated in every cycle.

COMPARE MATCH INTERRUPT

- A compare match interrupt is called when the value of the timer equals a specific value, set by the user.
- This value is set by setting the value of OCR register.
- Before incrementing, the value of the timer is compared to **OCR**. If the two are equal, a COMPARE MATCH interrupt is generated

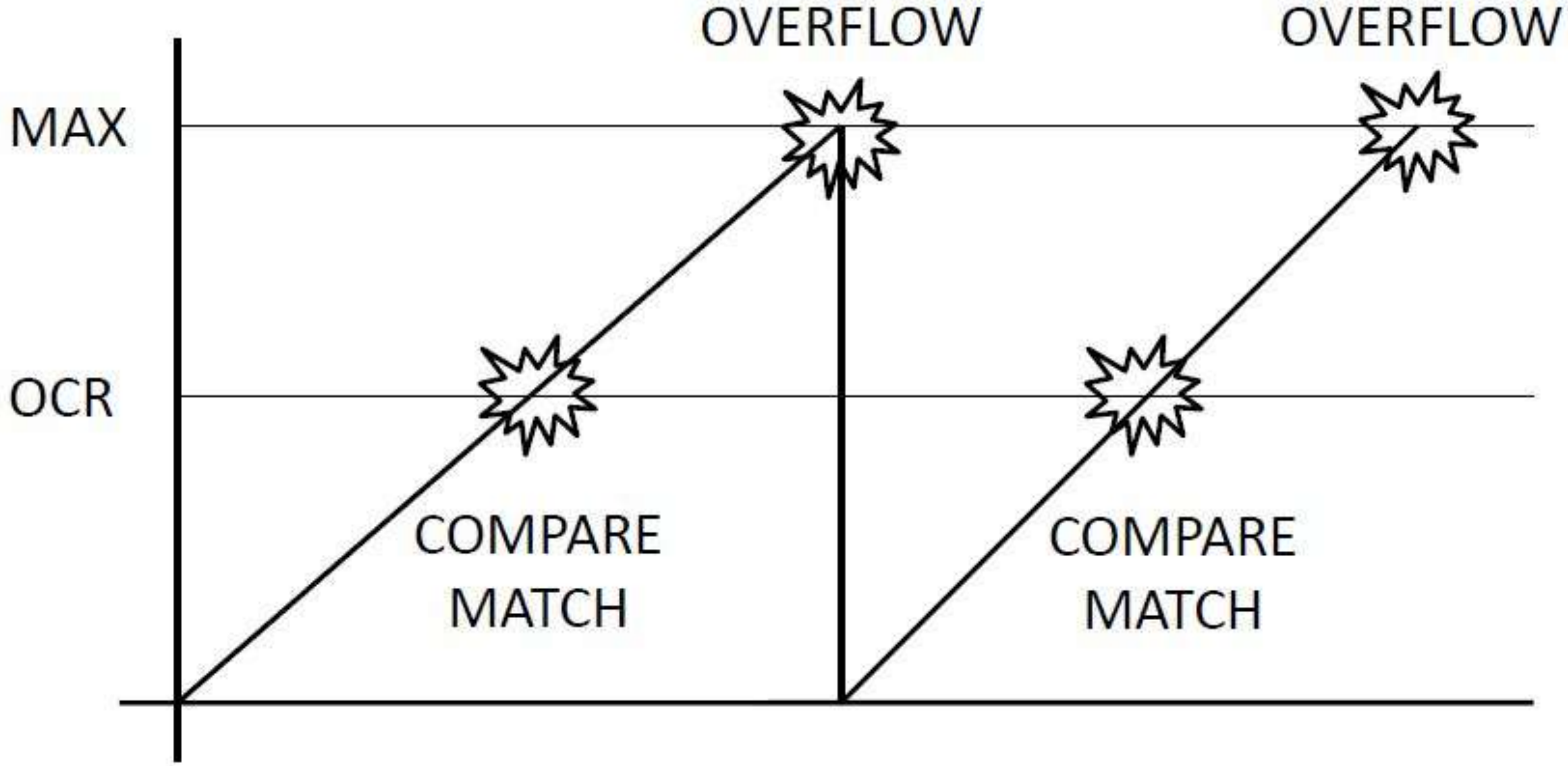
COMPARE MATCH STATISTICS

- Suppose a timer of maximum value ***n*** has a time period ***t*** (also called as clock period).
- Then :

1. Timer cycle frequency = $1/(n+1)\times t$
2. COMPARE MATCH interrupt frequency = $1/(n+1)\times t$

- If COMPARE MATCH interrupt is enabled, then an interrupt is generated in every cycle.

INTERRUPTS – OVERFLOW and COMPARE MATCH



Topics Covered so far…

- Registers
- Timers
- Interrupts
- External Interrupts
- Internal Interrupts
  - -- Overflow Interrupt
  - -- Compare Match Interrupt

TIMER MODES

- A timer works in three modes: Normal, CTC and PWM.
- All three modes differ in the response of the controller to the interrupts generated.
- The timer mode used so far in this presentation is normal mode.

Normal Mode

- Standard mode: Timer starts at 0, goes to maximum value and then resets itself.
- OVERFLOW and COMPARE MATCH interrupts generated as normal.

CTC Mode

- Known as Clear Timer on Compare.
- As evident by the name, the timer starts at 0 as usual, but instead of resetting after maximum value, it resets after reaching value specified in **OCR** register.
- Compare match interrupt if enabled will be generated but not overflow interrupt (Why?)

CTC Mode Statistics

- If clock time period is t:

1. Timer cycle time period $= (OCR+1)\times t$
2. Frequency $= 1/(OCR+1)\times t$

- With the use of CTC Mode we can theoretically generate any frequency up to 8 MHz.
- Example of 1 Hz generation.

Topics Covered so far…

- ✓ Registers
- ✓ Timers
- ✓ Interrupts
- ✓ External Interrupts
- ✓ Internal Interrupts

    -- Overflow Interrupt

    -- Compare Match Interrupt
- ✓ Timer Modes

    -- Normal Mode

    -- CTC (Clear on Timer Compare) Mode

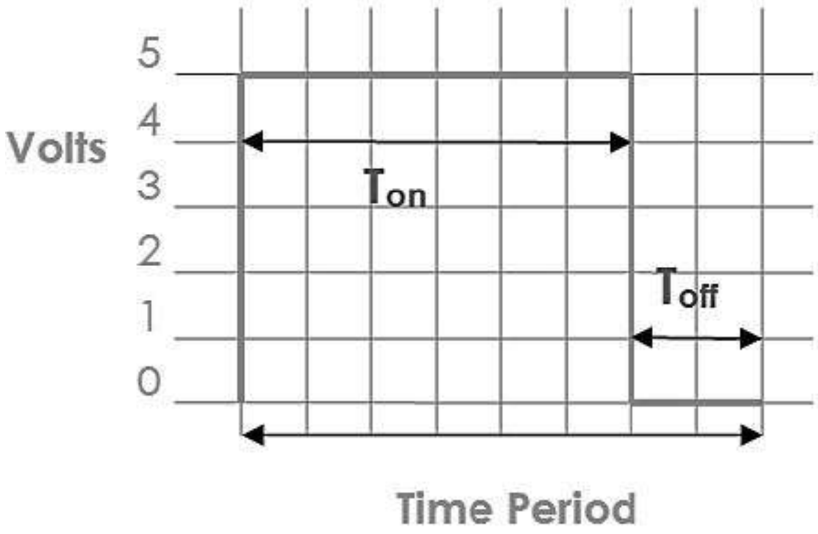
ELECTRONICS CLUB IITKANPUR

PWM Mode

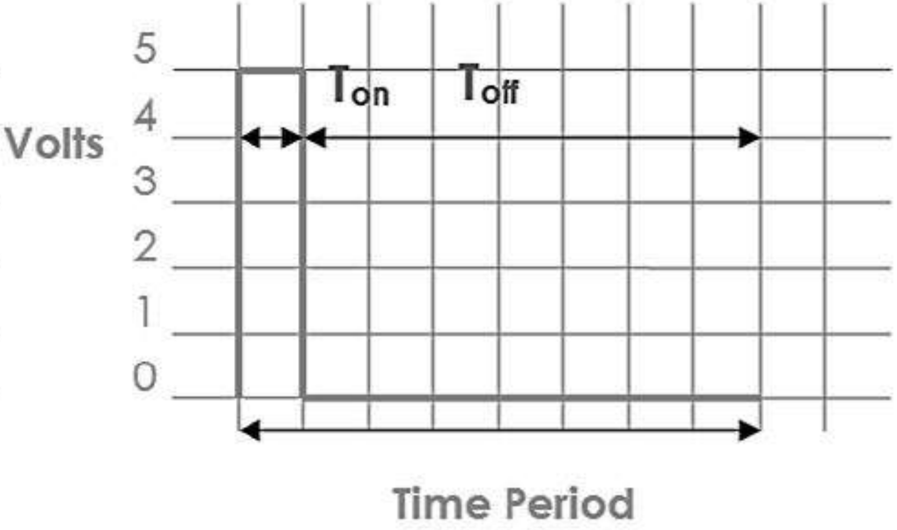
- Known as Pulse Width Modulation
- Simple method of obtaining analog output of any value between 0 and 5V.
- How is it achieved??

PWM Mode

- Suppose we need 3V for our device at a specified pin.
- We supply 5V on it for (3/5)* 100 % = 60% of the time period and 0V for the remaining time period
- The average voltage at the pin for a time period becomes 3V
- If this step is repeated very fast (T is very small), then the output behaves as a analog signal of 3V.



Vout = 3.75 V



Vout = 0.625 V

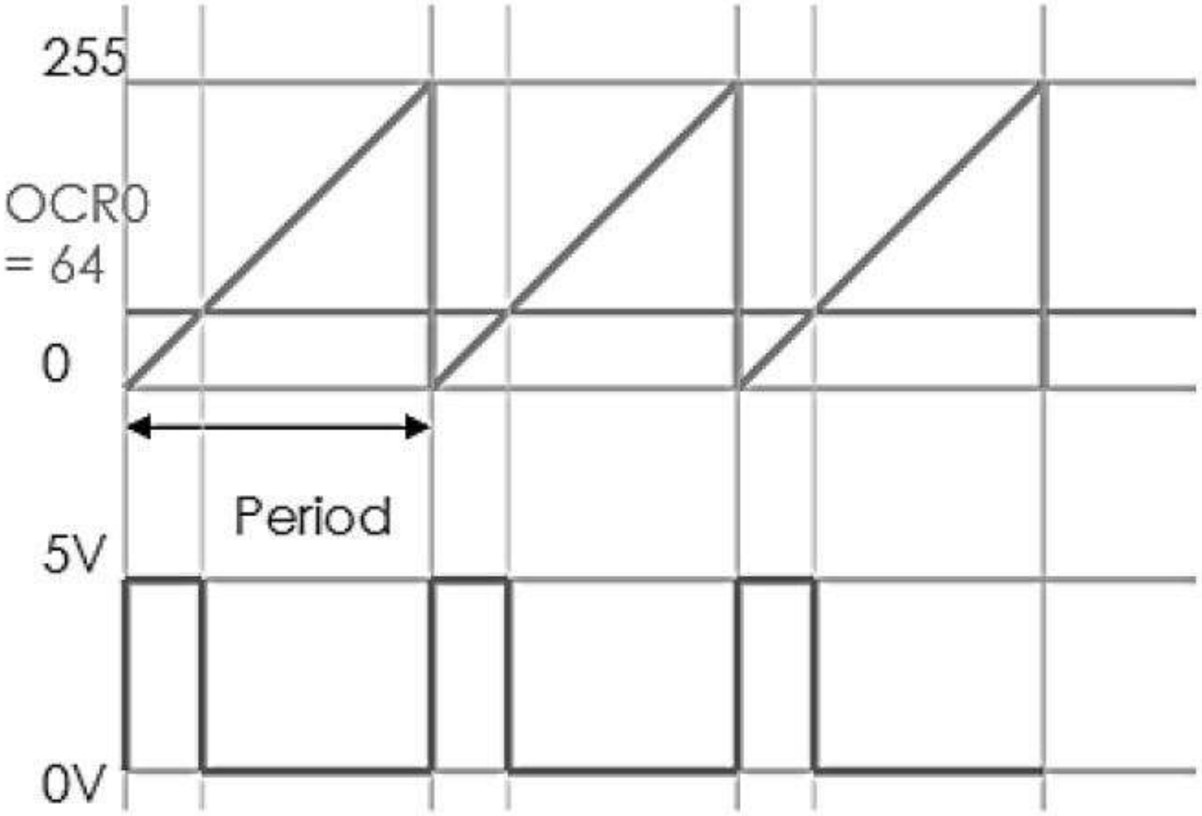
PWM Mode

- The PWM behaves in a similar way.

PWM Mode

- The PWM behaves in a similar way.
- This “analog” value is obtained using timers.
- A specific pin is set as output. When the timer reaches 0, the voltage of the pin is set to 5V.

PWM Mode



OC0 PIN

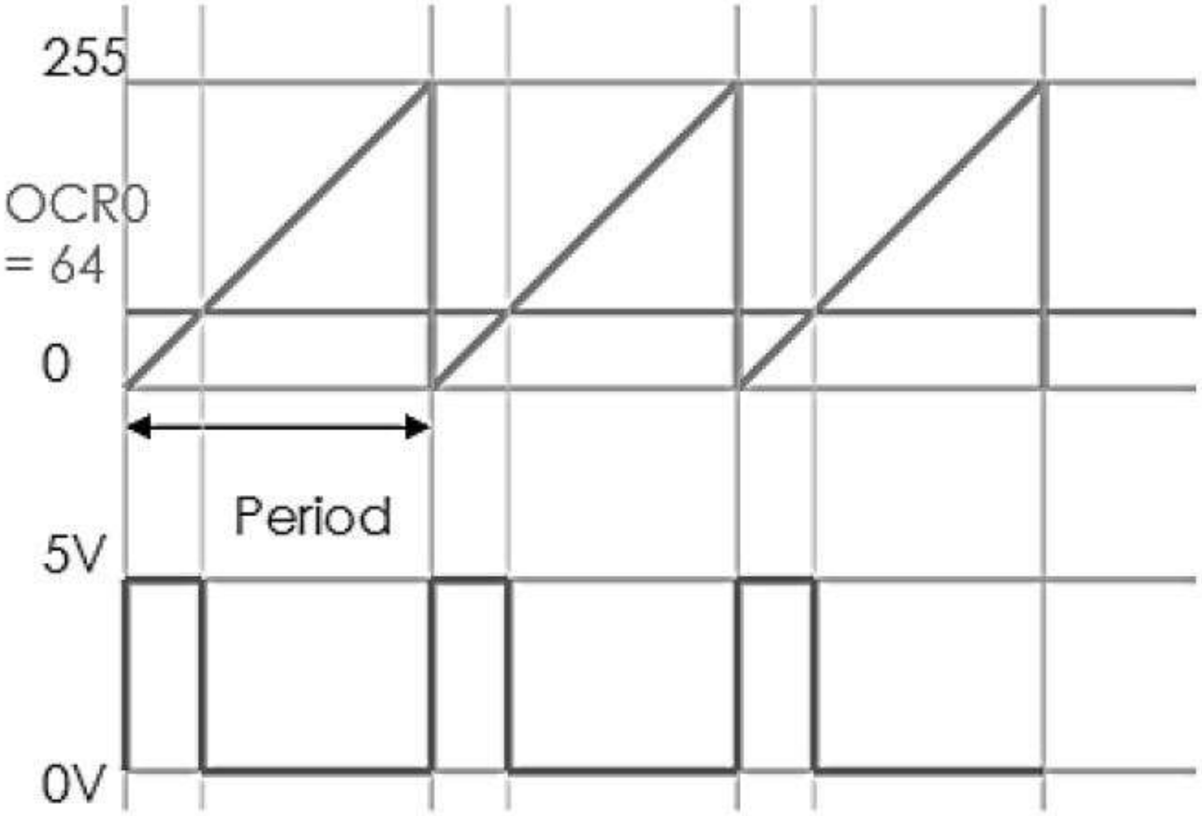
PWM Mode

- The PWM behaves in a similar way.
- This “analog” value is obtained using timers.
- A specific pin is set as output. When the timer reaches 0, the voltage of the pin is set to 5V.

PWM Mode

- The PWM behaves in a similar way.
- This “analog” value is obtained using timers.
- A specific pin is set as output. When the timer reaches 0, the voltage of the pin is set to 5V.
- When the timer reaches the value specified by OCR, on the next clock, the pin voltage is set to 0 until the timer resets itself.

PWM Mode



OC0 PIN

PWM Mode Statistics

- If clock time period is *t* and maximum timer value is *n*:

  1. Timer cycle time period $=(n+1)\times t$
  2. Frequency $=1/(n+1)\times t$
  3. Duty cycle $=[OCR/(n+1)]\times 100\%$
  4. Output voltage $=[OCR/(n+1)]\times 5V$

- COMPARE MATCH interrupt and OVERFLOW interrupt both will work properly.
- Demo.

Topics Covered so far…

- ✓ Registers
- ✓ Timers
- ✓ Interrupts
- ✓ External Interrupts
- ✓ Internal Interrupts
  - -- Overflow Interrupt
  - -- Compare Match Interrupt
- ✓ Timer Modes
  - -- Normal Mode
  - -- CTC (Clear on Timer Compare) Mode
  - -- PWM (Pulse Width Modulation) Mode

ELECTRONICS
EC IITKANPUR CLUB

Thank You…

ELECTRONICS
IITKANPUR CLUB